CIKM 2026 · Rome, Italy

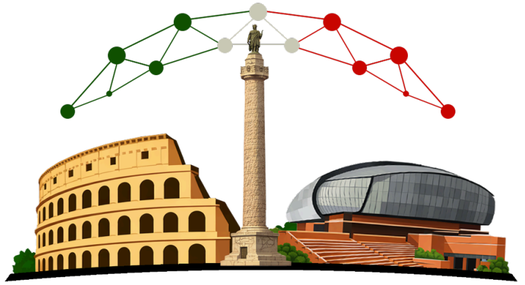
EPTQ

Fast and Accurate 2-bit Post-Training Quantization via Factored E_8 Lattice

Ha-Myung Park, Jungeun Park, Chaewon Jeong, Woncheol Son, Yunjun Kim, Junseo Youn, Ki Yeon Park

Kookmin University · hmpark@kookmin.ac.kr





CIKM 2026

Problem definition

Post-training quantization (PTQ) of LLM weights

LLMs are too big for one GPU — 2-bit PTQ makes them fit.

Too big for one GPU

Llama · GPT · Gemma · Qwen · DeepSeek

Llama-2-70B · FP16

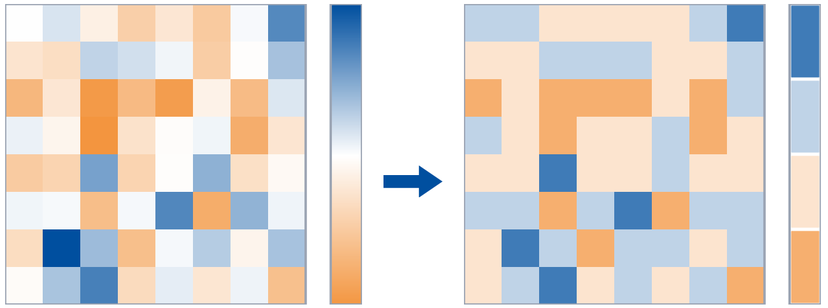
140 GB

RTX PRO 6000 · 96 GB

H100 · 80 GB

Almost all of it is the **weight matrices W**

Post-training quantization



W · 16-bit → **Ŵ · 2-bit**

$$\min \| WX - \hat{W}X \|^2$$

No retraining · small calibration input X

This work: 2-bit weights

16 → 2 bits

Llama-2-70B · FP16 → 2-bit

140 GB

≈ 18 GB (8× smaller)

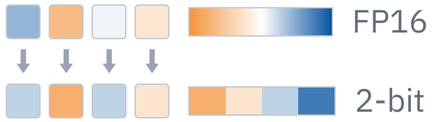
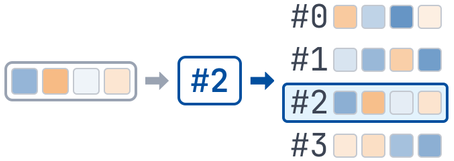
RTX 5090 (gaming) · 32 GB

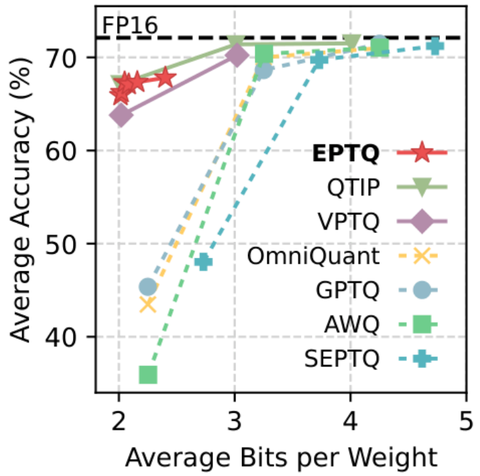
Fits on a gaming GPU

Motivation

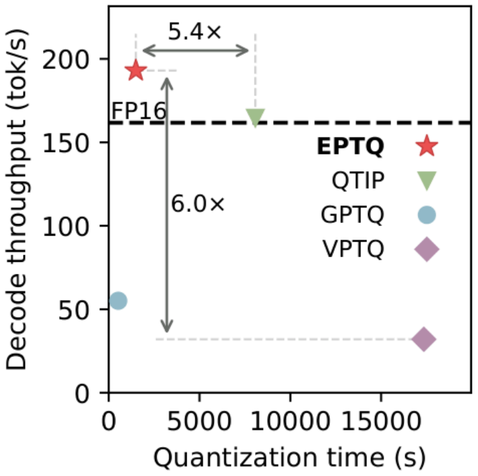
At 2 bits, accuracy and efficiency collide

Existing 2-bit methods are either accurate or efficient, not both.

	Scalar PTQ GPTQ, AWQ	**Vector PTQ** VPTQ, QTIP
	 each weight → its own level	 4 weights → 1 code → codebook vector
Accuracy	⚠ **Collapses** Avg acc. 72% → 45%	✓ **Good**
Efficiency	✓ **Fast**	⚠ **Slow** VPTQ 32 tok/s · QTIP 2.2 h to quantize



(a) Accuracy vs. bits
Llama-2-13B



(b) Quantization time vs. decode speed
Llama-3-8B

Top-left is better in both.

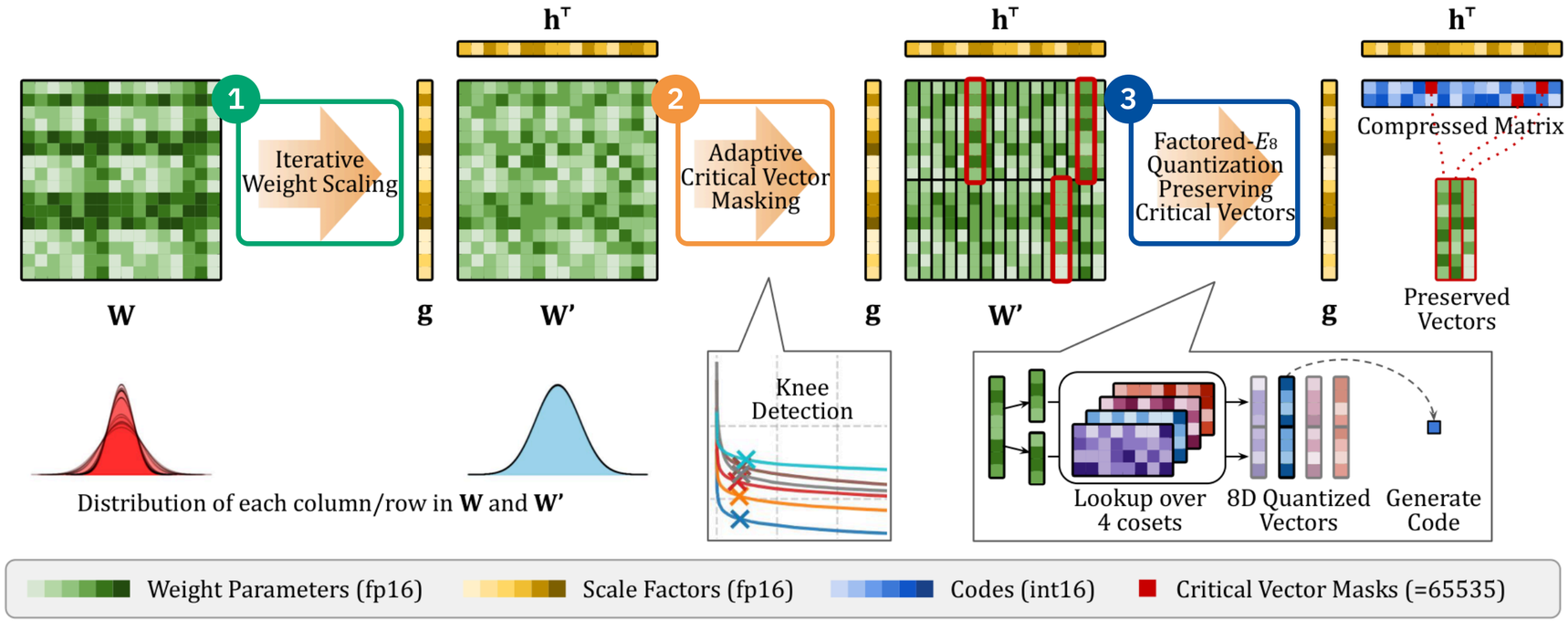
Can we have both?

Method

EPTQ at a glance

EPTQ gets both accuracy and speed with three components.

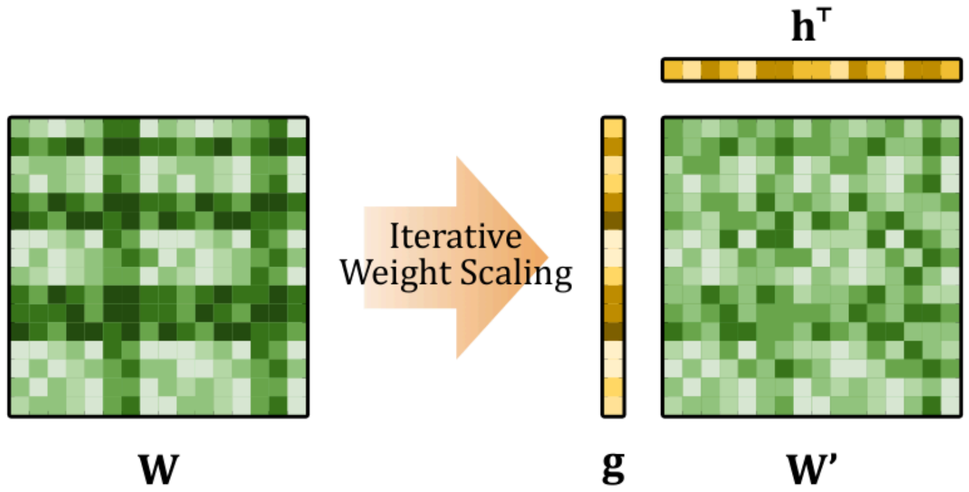
1 **Weight scale normalization** 2 **Adaptive critical weight preservation** 3 **Factored-E_8 quantization**



Method

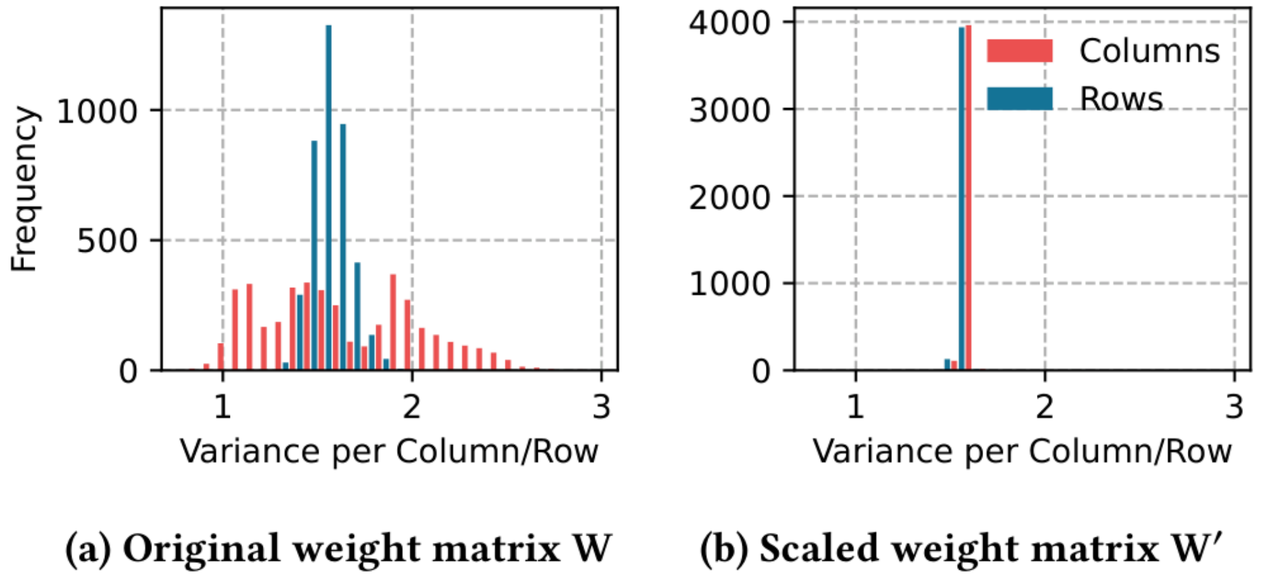
Weight Scale Normalization

We normalize the weight matrix with row and column scale vectors, g and h.



$$W = W' \odot (g\, h^{\mathrm{T}})$$

⊙ element-wise multiply



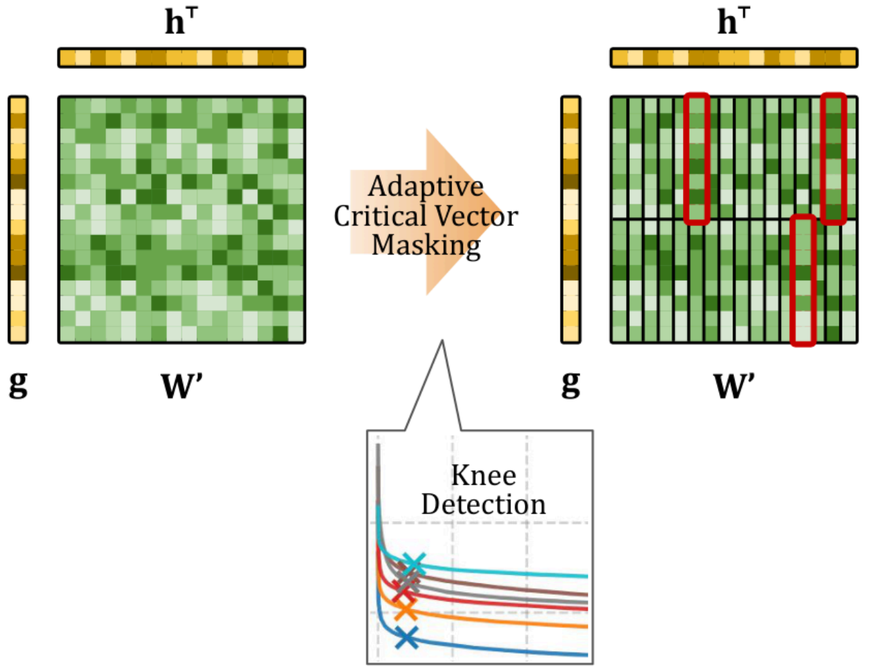
(a) Original weight matrix W (b) Scaled weight matrix W′

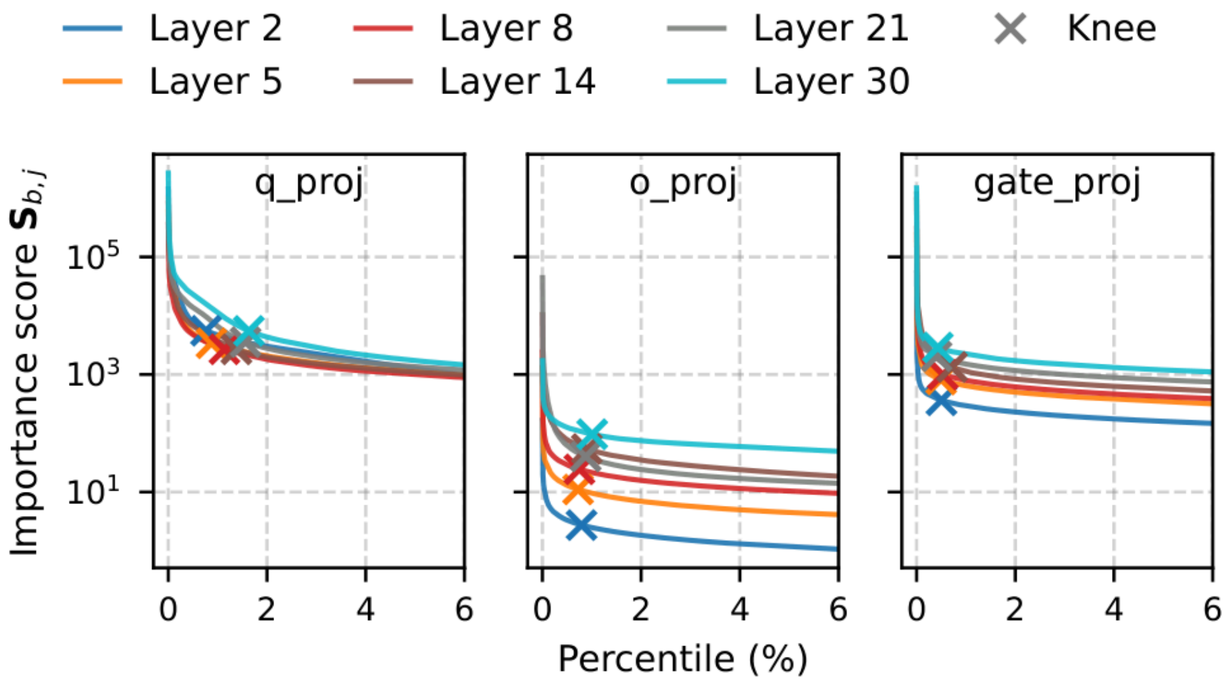
Sinkhorn-Knopp: divide by the row means, then the column means, and repeat.

Method

Adaptive Critical Weight Preservation

We keep a few critical vectors in FP16, and select them adaptively for each layer.



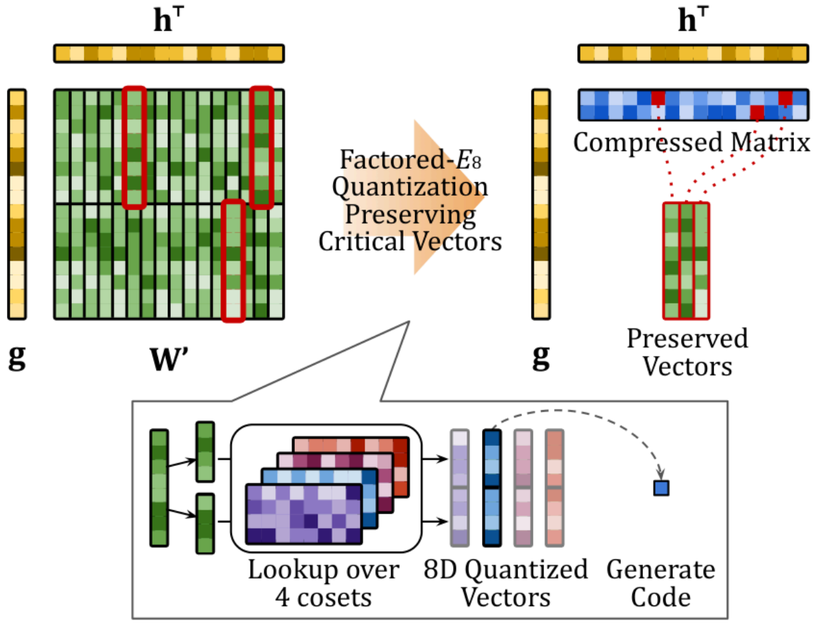


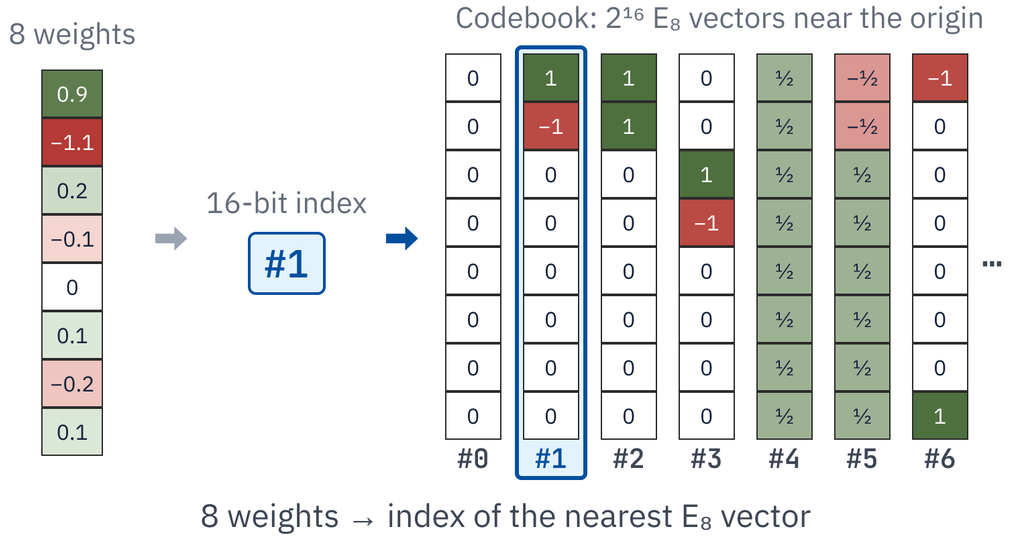
Kneedle: finds the knee point of the importance scores automatically.

Method

Factored-E_8 Quantization

We quantize every 8 weights into a 16-bit index of an E_8 codebook.





8 weights → index of the nearest E_8 vector

$$E_8 = \{\, x \in \mathbb{Z}^8 \cup (\mathbb{Z}+½)^8 : \Sigma x \text{ is even} \,\}$$

All integers or all half-integers, with an even sum

Why E_8?

Densest sphere packing in 8-D → small quantization error.

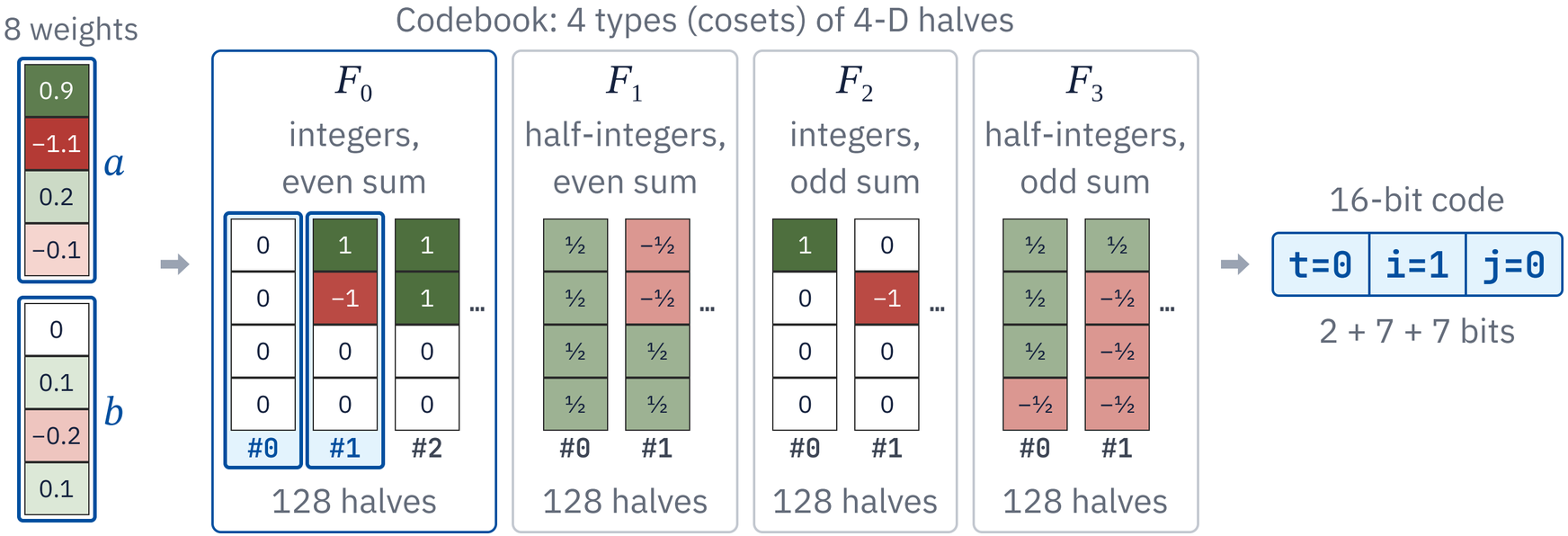
But stored directly

2^{16} vectors → 1 MB → does not fit in GPU L1 → slow lookup.

Method

Factored-E_8 Quantization

We store only 4-D halves, so the codebook shrinks from 1 MB to 4 KB.



Interesting point! Both halves always have the same type.

4 KB codebook

4 types × 128 halves → fits in GPU L1 → fast lookup.

Still exactly on E_8

$4 \times 128 \times 128 = 2^{16}$ points, all on the E_8 lattice.

On par with QTIP, far ahead of scalar PTQ

EPTQ is competitive with QTIP on Llama-2, but QTIP still leads on Llama-3-8B.

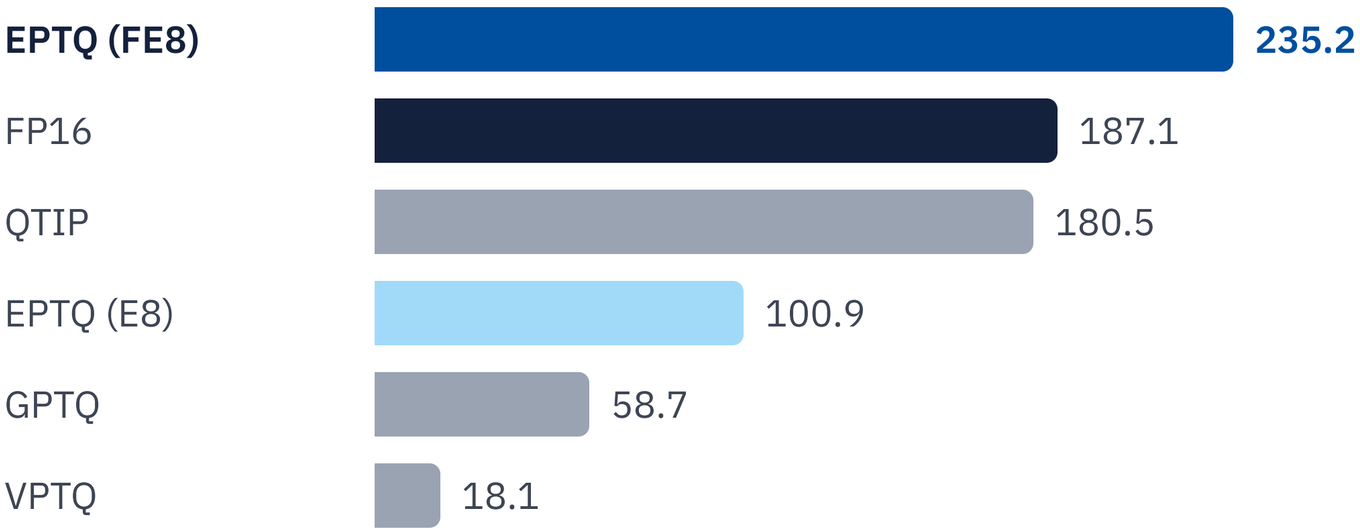
	Llama-2-7B		Llama-2-13B		Llama-3-8B	
Method (2-bit)	**Wiki2 ↓**	**Avg ↑**	**Wiki2 ↓**	**Avg ↑**	**Wiki2 ↓**	**Avg ↑**
FP16 (16-bit)	5.47	69.35	4.88	72.10	6.14	73.22
GPTQ	83.12	38.95	19.72	45.37	–	–
AWQ	222,150	35.90	122,749	35.92	–	–
VPTQ	9.12	59.23	7.22	63.80	13.01	60.30
QTIP	7.34	**63.52**	**6.10**	67.12	**9.98**	**64.30**
EPTQ (E8, ρ=0.1)	**7.28**	62.75	6.11	**67.53**	10.49	62.04
EPTQ (FE8, ρ=0.1)	7.45	62.72	6.26	67.12	11.10	60.86

Setup: Llama-2 7B/13B/70B and Llama-3-8B · 128 C4 calibration samples · one H100 · no fine-tuning. E8 = full 1 MB codebook, FE8 = Factored-E_8 (4 KB). Bold = best 2-bit result; GPTQ/AWQ on Llama-2 only.

Faster than FP16 at inference, and fast to quantize

EPTQ (FE8) decodes faster than FP16 and quantizes up to 6.8× faster than QTIP.

Decode throughput, Llama-2-7B, ρ = 0.02 (tok/s)



Llama-3-8B: EPTQ (FE8) 193.0 vs. FP16 161.8 and QTIP 164.7 tok/s

6.4×

faster quantization than QTIP on Llama-2-13B (2,115 s vs. 13,531 s)

6.8×

on Llama-2-70B (2.3 h vs. 15.9 h)

Both scaling and adaptive preservation help

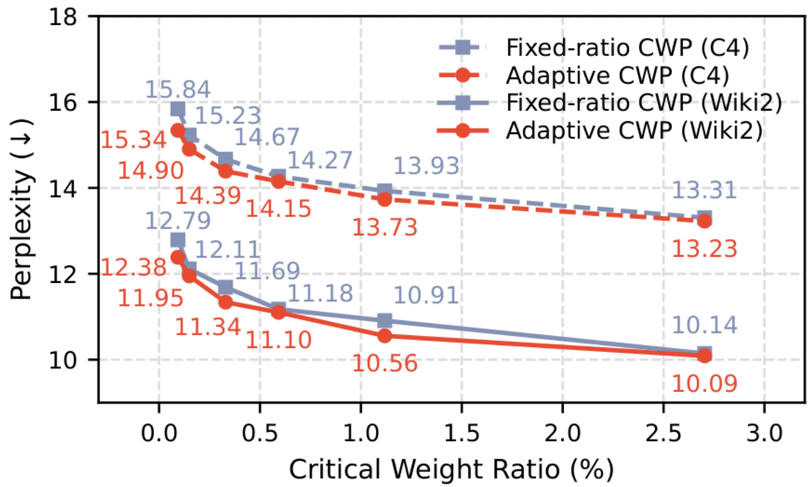
Scale normalization and adaptive preservation both lower perplexity.

Adding weight scale normalization (FE8)

Model	ρ	Δ C4 PPL ↓	Δ 0-shot ↑
Llama-2-7B	0.02	−1.44	+3.0
Llama-2-7B	0.1	−0.66	+1.7
Llama-2-13B	0.02	−0.59	+1.5
Llama-2-13B	0.1	−0.37	+0.9
Llama-3-8B	0.02	**−3.44**	**+7.3**
Llama-3-8B	0.1	**−2.33**	**+4.2**

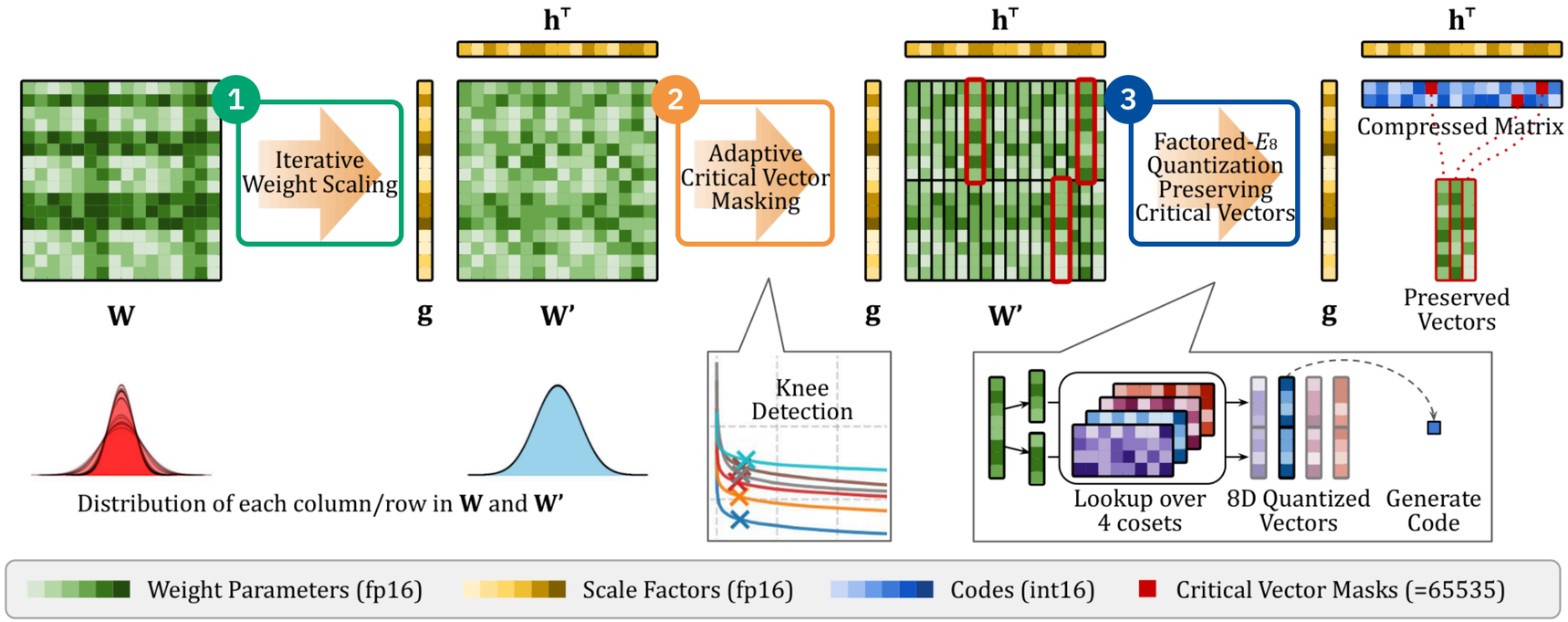
Improves every model and ρ; largest on Llama-3-8B.

Adaptive vs. fixed-ratio, Llama-3-8B



Same critical-weight budget, lower PPL at every ratio.

EPTQ: Fast and Accurate 2-bit Post-Training Quantization via Factored E_8 Lattice



Thank you · Questions?

Code and models: github.com/kmudmlab/eptq

Ha-Myung Park

hmpark@kookmin.ac.kr · hmpark.me

Data Mining Lab, Kookmin University

dm.kookmin.ac.kr